МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Физтех-школа физики и исследований им. Ландау

Машинное обучение. Конспект

Автор:
Яренков Александр Владимирович

Долгопрудный
18 июля 2024 г.

1 Введение

Определим основные понятия и покажем их на примере из таблицы под этим предложением.

Name	Age	Statistics (mark)	Python (mark)	Eye color	Native language	Target (mark)	Predicted (mark)
John	22	5	4	Brown	English	5	1
Aahna	17	4	5	Brown	Hindi	4	5
Emily	25	5	5	Blue	Chinese	5	2
Michael	27	3	4	Green	French	5	4
Some student	23	3	3	NA	Esperanto	2	3

- Датасет/выборка/набор данных - данные, которые нам доступны

  Вся таблица, кроме последней колонки.

- Объект/наблюдение/datum/data point - элемент выборки

  Обычно(всегда ли?) мы предполагаем их независимыми и одинаково распределёнными (н.о.р.).

  Строка в таблице

- Признак/feature - характеристика объекта.

  Все колонки признаков образуют матрицу признаков/design matrix (X) (её столбцы x_l - вектора, имеющие все признаки объекта).

  Первые 6 колонок в таблице

- Целевая (зависимая) переменная/target (y) - переменная, значение которой нас интересует

  В задаче бинарной классификации (target принимает значения из множества мощностью 2) имеет специфическое название - label. Задача регрессии ($y \in \mathbb{R}$)

  Жёлтый столбец в таблице

- Модель - метод обучения. Формально, действующее на признаковом пространстве отображение ($f(X)$)

- Предсказание/prediction ($\hat{y}$) - результат работы модели

  Зелёная колонка в таблице

- Функция потерь/Loss (L) - критерий качества модели. Её значение мы пытаемся уменьшать, чтобы модель была качестсвеннее.

  Примеры функции потерь:

  Mean Squared Error/дисперсия: $MSE(y, \hat{y}) = \|y - \hat{y}\|_2 = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

  Mean Absolute Error: $MAE(y, \hat{y}) = \|y - \hat{y}\|_1 = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$

  Здесь $N-$ количество наблюдений

- Параметры - Переменные, значение которых модель выбирает сама

  Например, взвешивание данных

- Гиперпараметры - параметры, которые мы фиксируем до начала работы с данными

  Например, глубина решающего дерева

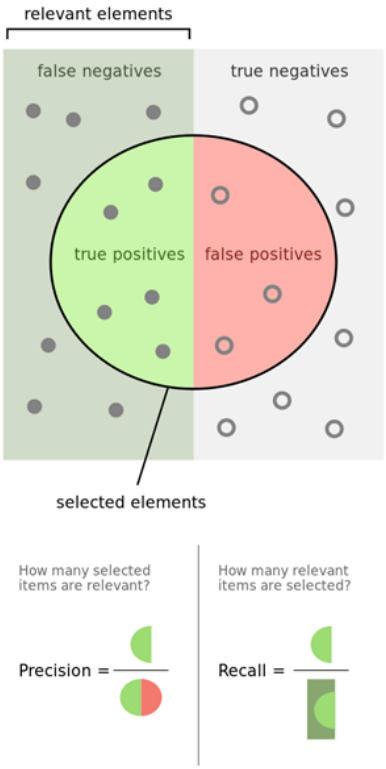
1.1 Формальная задача обучения с учителем

Обучение с учителем - обучение с наличием целевой переменной.

Поставим формально задачу обучения с учителем: $\{x_i, y_i\}_{i=1}^N$ – тренировочный датасет(training set), $f(X)$ – модель f, которой на вход подаём матрицу признаков X и получаем предсказание f(X) $= \hat{y}$, $L(X, y, f)$ – Loss function, that should be minimized

1.2 Метрики

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive** TP	**False positive,** FP Type I error
	Predicted condition negative	**False negative,** FN Type II error	**True negative** TN



Precision - доля верно классифицированных объектов (TP) ко всем классифицированным объектам (TP + FP). Она же точность

Recall - доля верно классифицированных объектов (TP) ко всем объектам этого класса (TP + FN). Т.е. мы узнаём сколько значений из класса мы НЕ потеряли.

Есть задачи в которых важнее либо precision, либо recall. Например, заболевшие люди приходят в больницу. Если им назначить лечение, не факт что им станет лучше. Но если им не назначить лечение, то им может стать сильно хуже. Поэтому нам не так важно улучшить состояние каждого (повысить precision - в данном случае доля заболевших людей, у которых улучшилось состояние), как чтобы людям не становилось ещё хуже (т.е. нужно повысить recall - в данном случае доля заболевших людей, у которых не ухудшилось состояние)

Поэтому введём понятие F_β–score. Это метрика, учитывающая важность либо precision, либо recall.

$$F_\beta = (1 + \beta^2)\frac{Precision \cdot Recall}{\beta^2 Precision + Recall}$$

Наиболее частые частные случаи:

$\mathrm{F}_\beta|_{\beta=0} = F_0 = 2 \cdot Precision$
$\mathrm{F}_\beta|_{\beta=+\infty} = F_\infty = Recall$
$\mathrm{F}_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$

2 Наивный байесовский классификатор

$\{x_i, y_i\}_{i=1}^N$ - training set. $x_i \in \mathbb{R}^p$ - вектора, $y_i \in \{C_i\}_{i=1}^k$ (Задача классификации на k классов).

А также наивное предположение, которое делает байесовский классификатор наивным: все признаки независимые.

Согласно теореме Байеса,

$$P(y_i = C_j|x_i) = \frac{P(x_i|y_i = C_j) \cdot P(y_i = C_j)}{P(x_i)} \quad (1)$$

Так как все признаки считаем независимыми, то происходит следующая факторизация:

$$P(x_i|y_i = C_j) = \prod_{l=1}^{p} P(x_i^l|y_i = C_j)$$

Таким образом вероятность $P(x_i|y_i = C_j)$ мы поняли как считать. Вероятность $P(y_i = C_j)$ мы сможем посчитать по частоте встречаемости экземпляров класса в тренировочной выборке.

Так как у нас стоит задача классификации, то ответом к ней будет класс, к которому относится объект: $\boxed{C = \arg\max_j P(y_i = C_j|x_i)}$ - аргумент, при котором вероятность максимальна. Заметим, что максимизируем мы по различным классам, а знаменатель выражения (1) не зависит от j, поэтому для задачи максимизации он нам не нужен и мы его просто выбросим из рассмотрения. При желании его посчитать, можно воспользоваться формулой полной вероятности.

3 Градиентная оптимизация

3.1 Безусловная оптимизация

Пусть модель зависит от параметров w: $f_w(X)$. Наша задача в повышении качества нейронной сети. Математически это качество выражается в виде функции качества, или обратной к ней - функции потерь. Т.о. наша задача состоит в минимизации функции потерь.

Для этого подберём параметры модели, при которых функция потерь будет наименьшей (**это и есть оптимизация**). Так как градиент - это направление наискорейшего роста, то будет подбирать параметры модели следующим образом:

$$w_{n+1} = w_n - \alpha \nabla_w L(y, f_w(X))$$

где (w_n) - последовательность параметров модели (при $n \to \infty$ последовательность должна сходиться к локальному минимуму), w_0 - начальные значения параметров, α - шаг оптимизации/learning rate - некоторый коэффициент, который может меняться на каждом шагу.

3.2 Условная оптимизация

Условная оптимизация - это оптимизация параметров, на которые наложены ограничения.

Например, в нашей задаче есть n классов и в каждом из них какое-то количество объектов. Вероятность посмотреть на элемент $i_{\text{того}}$ класса равна p_i. Эти вероятности и будем считать параметрами. На них есть понятное ограничение: $\sum_{i=1}^{n} p_i = 1$. Это простой случай, т.к. это ограничение со знаком равенства, а значит будем применять метод множителей Лагранжа (3й семестр мат. анализа) для нахождения условных экстремумов.

В качестве основы для функции Лагранжа выберем энтропию (*почему?*) $S = -\sum_k p_k \ln p_k$

Тогда функция Лагранжа $\mathcal{L} = -\sum_k p_k \ln p_k + \lambda(1 - \sum_k p_k)$

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial p_i} = -\ln p_i - 1 + \lambda = 0 \\ \\ 1 - \sum\limits_k p_k = 0 \end{cases}$$

$$\begin{cases} \ln p_i = \lambda - 1 => p_i = \exp(\lambda - 1) \\ \\ 1 - \sum\limits_k p_k = 0 => 1 - \sum\limits_k \exp(\lambda - 1) = 0 => 1 - k\exp(\lambda - 1) = 0 => \lambda = 1 - \ln k \end{cases}$$

Таким образом $\ln p_i = -\ln k$ или $p_i = \frac{1}{k}$. Мы нашли кандидатов на условный экстремум и дальше делаем с ними тоже самое, что и для градиентной оптимизации.

4 Методы регрессионного анализа

4.1 Линейные модели и линейная регрессия

Модель $f(X)$ назвается линейной, если она является линейной комбинацией признаков и какого-то числа.

Поставим формально задачу линейной регрессии: $\{x_i, y_i\}_{i=1}^N -$ training set , $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$

$f(X) = \omega_0 + \sum\limits_{k=1}^{p} \omega_k x_k$

Свободный член w_0 называют **bias term**, т.е. параметр смещения. Потому что в линейной функции свободный член отвечает за смещение прямой относительно начала координат.

Эта запись выглядит довольно громоздко. Её можно упростить с помощью скалярного произведения признаков на веса. Но тогда останется bias term w_0. Поэтому, чтобы писать только скалярное произведение введём следующие обозначения.

Обозначим $\tilde{X} = (1, x_1, x_2, ... x_p)^T$, а $W = (w_0, w_1, w_2, ... w_p)^T$ - вектор весов. Тогда $f(X) = \tilde{X}^T W$

Для решения задачи нам нужно найти такие веса W, которые минимизируют функцию потерь. Эти искомые веса $\hat{W} = \arg\min\limits_w L$. Тогда решением задачи/предсказанием модели будет $\hat{Y} = XW$

Обычно задачи машинного обучения не имеют аналитического решения, но если мы выберем функцию потерь $L = \|Y - XW\|_2^2$ и будем её минимизировать, то по сути мы будем применять МНК.

$L = (Y - XW)^T(Y - XW)$

$\nabla_w L = \nabla_w(Y^TY + W^TX^TXW - Y^TXW - W^TX^TY) = 0 + X^TXW \cdot 2 - X^TY \cdot 2 = 0 =>$
$W = (X^TX)^{-1}X^TY$. Это решение существует, только есть матрица X^TX обратима, т.е её детерминант не равен 0, т.е все признаки линейно независимы. Есть даже **Теорема Гаусса-Маркова**, которая утверждает что если $Y = XW + \varepsilon$, где $\varepsilon - (\forall\ i\ \mathbb{E}\varepsilon_i = 0)$, дисперсии конечны, а $\forall$ i, j $cov(\varepsilon_i, \varepsilon_j) = 0$, то решение найденное по МНК - оптимальное несмещённое решение.

Но что делать в случае когда признаки зависимые?

4.2 Регуляризация

Т.к. признаков стало меньше, то для сохранения единственности решения необходимо ввести дополнительные ограничения. Такое введение ограничений называется регуляризацией.

Например, можно добавить диагональную матрицу для получения линейной независимости и самый простой способ - добавить единичную матрицу. $W = (X^TX + \lambda E)^{-1}X^TY$. На самом деле

такой вектор весов это будет решение с помощью МНК для следующей функции потерь: $L = \|Y - XW\|_2^2 + \lambda^2\|W\|_2^2$. Это есть L_2 регуляризация. Частный случай L_p регуляризации.

Вообще L_2 регуляризация пытается найти минимум используя все признаки, т.к. имеет меняющий знак в нуле линейный градиент, а L_1 регуляризация пытается занулить наибольшее количество весов, т.к. имеет меняющий знак в нуле константный градиент.

4.3 Задача классификации

Введём понятие правдоподобия. Пусть наша модель работает с параметрами θ. Тогда функция правдодобия - это вероятность $P(X,Y|\theta)$ пронаблюдать такие X и Y при параметрах θ. Логично, что функцию правдоподобия нужно максимизировать. Если X и Y одинаково распределённые $\forall i$ x_i и y_i независимые, то $P(X,Y|\theta) = \prod_i P(x_i,y_i|\theta)$. Далее мы можем воспользоваться монотонной функцией логарифм, т.е. максимизация логарифма величины это та же задача, что и максимизация самой величины. Тогда мы будем рассматривать $\sum_i \ln P(x_i,y_i|\theta) \to \max_\theta$

4.4 Логистическая регрессия

Регрессия называется логистической, т.к. есть "0"и "1". Соответсвенно имеем задачу бинарной классификации. Пусть $p \in [0,1]$ - вероятность принадлежности к одному из классов. Но мы пока умеем решать только задачу регрессии, и предсказание является элементом $\mathbb{R}$. Поэтому нам нужно провести какие-то преобразования с вероятностью p, такие что сохранится монотонность вероятности и новое выражение будет принимать значение из $\mathbb{R}$.

Один из способов следующий:

$$p \to \frac{p}{1-p} \in [0,\infty) \to \ln\frac{p}{1-p} \in \mathbb{R}$$

И уже это выражение можно приравнять к предсказанию линейной регрессии $f(X)$ (см. 4.1). Выражая отсюда p, получим $p = \frac{1}{1-e^{-f(X)}}$. Полученная функция называется **сигмоида**. $\sigma(f(X))$

Для задачи бинарной классификации определим понятие *Margin* (сокр. M, по-русски отступ).Margin показывает насколько глубоко в собственном классе находится точка. Введём каждому из двух классов метку: положительную 1 или отрицательную -1. Тогда $M =$ метка класса $\cdot f(X)$.

Итак, вероятность принадлежности к одному из классов, согласно выкладкам выше, есть $\sigma(f(X)) = \sigma(M)$. Тогда вероятность принадлежности к другому классу *с отрицательной меткой* есть $1 - \sigma(f(X)) = \sigma(-f(X)) = \sigma(M)$.

Таким образом, для максимизации функции правдоподобия, нам нужно максимизировать $\sum_i \ln P(x_i,y_i|\theta) = -\sum_i \ln\big(1 - \exp(-M_i)\big)$ или $\sum_i \ln\big(1 - \exp(-M_i)\big) \to \min_w$. Последнюю называют *логистической функцией потерь*, которую минимизируют.

4.5 Многоклассовая классификация

Чаще всего пользуются следующим методом: с помощью логистической регрессии относят данные к каждому классу по отдельности. Т.е. сколько классов, столько логистических регрессий и нужно будет построить. А потом логично с помощью биссектрис (равноудаление от линий, полученных в результате регрессии) углов добавить оставшиеся данные.

5 Метод опорных векторов (SVM)

5.1 Линейный

Начнём с простой выборки из двух классов, в которой гиперплоскость разделяет классы с нулевой функцией потерь, где в качестве функции потерь мы выбираем эмпирическую функцию ошибок, т.е. количество неправильно классифицированных точек. Пусть это будут, опять же, положительный и отрицательный класс с соответсвующими метками классов, тогда margin всегда будет положительный.

Отнормируем веса линейной регрессии таким образом, чтобы $\min\limits_i M_i = 1$. Будем здесь считать, что $W \in \mathbb{R}^n$. Это значит, что для любых точек $|W \cdot X - w_0| \geqslant 1$. В этом неравенстве мы по сути берём проекцию на нормаль, а значит в нём заключён margin. Ведь по своему смыслу margin это глубина нахождения точки в классе и мы хотим максимизировать его для получения наиболее хорошей модели. Поэтому найдём проекцию на нормаль ближайшей точки с одной стороны и ближайшей точки с другой стороны. Одна из проекций получится отрицательной, т.к. будет смотреть против нормали. Тогда вычитая одно из другого получим $\frac{(x_+ - x_-)\cdot W}{\|W\|} \geqslant \frac{2}{\|W\|} \to \max$. Видим, что эта оценка, которую мы максимизируем, не зависит от положения точек. А значит изменяя норму W, мы меняем и расстояние между классами.

Теперь формализуем задачу:

$$\begin{cases} \frac{\|W\|^2}{2} \to \min\limits_{W,w_0} \\ M_i \geqslant 1 \end{cases}$$

Теперь обобщим задачу на случай, когда невозможно провести линейную регрессию с нулевым лоссом. В таком случае margin должен уменьшиться (в противном случае это только усиливает рафинированность задачи). Поэтому добавим в формулировку $\xi_i \geqslant 0$. $M_i \geqslant 1 \ \to M_i \geqslant 1 - \xi_i$. Так как мы хотим совершать наименьшее число ошибок, т.е. минимизировать лосс, то и туда нужно добавить сумму всех этих ξ_i. Итого получится следующая формулировка задачи:

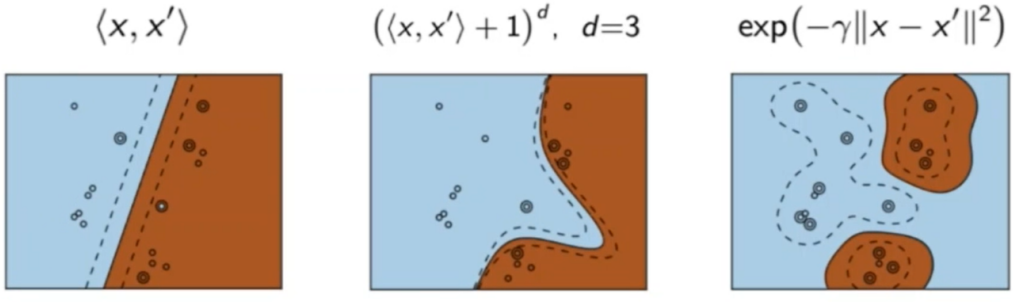
$$\begin{cases} \frac{\|W\|^2}{2} + C\sum\limits_{i=1}^{l} \xi_i \to \min\limits_{W,w_0,\xi} \\ \forall i \ M_i \geqslant 1 - \xi_i \\ \forall i \ \xi_i \geqslant 0 \end{cases}$$

Выражая отсюда ξ_i и убирая его получим следующую более простую формулировку задачи:

$$\frac{\|W\|^2}{2} + C\sum_{i=1}^{l}(1 - M_i) \to \min_{W,w_0}$$

Таким образом, получим так называемый *Hinge loss*: $L = \sum\limits_{i=1}^{l}[M_i < 0] \leqslant \sum\limits_{i=1}^{l}(1 - M_i) + \frac{1}{2C}\|W\|^2$

5.2 Нелинейный



Если мы поменяем скалярное произведение, то изменим метрическое пространство (матрица Грама). И на самом деле мы будем пользоваться тем же самым линейным SVM и будем получать гиперплоскости, но уже в другом пространстве, признаковом пространстве.

6 Оценка качества классификации

6.1 Коэффициент детерминации

$$SSE = \sum_i (y_i - \hat{y}_i)^2$$

$SST = \sum_i (y_i - < y >)^2$ Тогда определим коэффициент детерминации как $R^2 = 1 - \frac{SSE}{SST}$. Из формулы видно, что чем лучше наша модель, тем ближе коэффициент детерминации к 1. Однако, чем больше данных у нас будет, тем меньше будет каждое слагаемое в SST, а значит R^2 тоже будет увеличиваться. Таким образом, сравнивать качества моделей с разным количеством данных с помощью коэффициента детерминации некорректно.

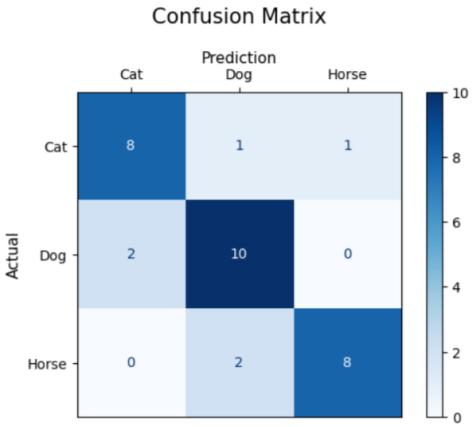
6.2 ROC-кривая

Модели машинного обучения могут создаваться для разных целей. Например, для определения болен человек или здоров. И эта модель может выдавать какое-то число (как в случае регрессии). Для примера, это может быть число от 1 до 10, где выше 7.5 - болен. Это число 7.5, по которому меняется отнесение объекта к классу называется *порогом*. И при разных порогах, очевидно, мы будем получать разные Presicion и Recall (взять банально классификатор, относящий все объекты только к одному классу). Так вот меняя этот самый порог, мы будем получать различные точки на Precision-Recall диаграмме. В нашем примере в случае порога равного 0, всегда предсказывается что человек болен. А в случае порога равного 10, всегда предсказывается что человек здоров. В случае чистой линейной регрессии без дальнейших преобразований это пороги $+\infty$ и $-\infty$.

Таким образом, мы получаем ступенчатую ROC-кривую в координатах Precision-Recall, всегда выходящую из 0, и заканчивающуюся в точке (1, 1). При увеличении количества точек эта кривая будет лишь сглаживаться, т.е. AUC (area under curve) должен остаться примерно тем же. Поэтому для оценки качества классификатора используют значение ROC-AUC. Чем оно выше, чем лучше классификатор. Если ROC-AUC $= 0.5$, т.е. количество отнесений объектов к классам не меняется, то логично что это классификатор, который совершенно случайно даёт предсказания. Т.е бесполезный :)

6.3 Матрица ошибок (Confusing matrix)

Матрица ошибок - это квадратная матрица размера $n \times n$, где n - количество классов. Эта матрица показывает количество правильно и неправильно классифицированный объектов. Вот пример, как может выглядеть матрица ошибок.



7 Cross-validation

Обычно выборку данных делят на 3 подвыборки: train, valid, test. На train выборке модель обучается, на valid выборке проверяется качество обученной модели Обычно valid выборка является составляющей train выборки. Затем на test выборке, которую модель никогда до этого не видела (в отличие от valid) окончательно проверяется качество обученной модели.

Кросс-валидация - это когда мы берём разные в train подвыборки (valid) несколько раз, тем самым обучаясь на train − valid и валидируясь на valid каждый раз на разных поднаборах данных. Например, можно поделить выборку на 5 частей и обучить модель 5 раз, валидируясь на каждой из 5 частей по очереди и, соотвественно, обучаться на оставшихся 4 частях.

Есть и более сложные методы кросс-валидации. Например, можно каждый раз брать по какому-то "кусочку"из 5 частей из примера выше так, чтобы в сумме на валидации была $\frac{1}{5}$ train набора данных. Это называется стратифицированной выборкой. А можно вообще случайным образом выбирать valid подвыборку из train.

Однако так не получится делать, если данные на которых предстоит обучаться машине это временные ряды. Обучаться на будущем и предсказывать прошлое не имеет смысла.

8 Решающие деревья

Решающее дерево - это набор условий, по которым разделяются данные в разные классы. Всё начинается с корня дерева - начального условия. Затем по веткам переходим либо в новые условия и ветвимся дальше, либо попадаем в конец дерева (лист), где уже определяется класс.

Осталось понять как выбирать эти условия. Пусть мы дошли до ветки с выборкой Q, которая ветвится на выборки L и R. Пусть H - критерий по которому мы будем выбирать условие. Этот критерий выбирается так, что чем меньше его значение, тем предпочительнее будет выбрать такое условие. Тогда нам нужно будет просчитать этот критерий для каждого возможного условия в подвыборках L и R и, так как эти выборки могут быть разного размера, это необходимо учитывать. По итогу, мы должны минимизировать следующее выражение на каждом ветвлении дерева:
$\frac{|L|}{|Q|}H(L) + \frac{|R|}{|Q|}H(R)$.

Какие же бывают критерии? Для задачи регрессии можно взять, например, MSE. Для задачи классификации 2 наиболее известных это критерий информативности и критерий Джини. С первым всё понятно, мы используем статистическую энтропию, у которой минимум в вырожденном распределении и максимум в равномерном. Первое легко показать, а второе было доказано ранее в 3.2. Энтропию, как меру хаоса, мы стараемся свести к минимуму.

Теперь о критерии Джини. При выборе с возвращением вероятность взять одно и то же данное 2 раза обозначим p_k^2. Вероятность не вытащить 2 данных из одного класса соответственно $1 - p_k^2$. Пусть у нас N классов. Тогда вероятность того, что мы не вытащим 2 подряд данных из одного класса $G = 1 - \sum_{k=1}^{N} p_k^2$. Это, как и энтропия, ещё один способ измерить разнородность данных.

Решающие деревья можно усекать (англ. pruning). Можно это делать до построения дерева, например, ограничить максимальную глубину дерева или размер листа (количество элементов в нём). Либо уже усечь уже обученное дерево с минимальными потерями качества модели. Это нужно для того, что решающие деревья очень склонны к переобучению и их нужно усекать.

Решающие деревья обладают ещё одним полезным свойством, а именно что на пропуски в данных можно забить. Если про какой-то признак у некоторых данных у нас нет информации, то мы можем эти данные отправить в обе ветки, посчитать взвешенное среднее предсказаний и тогда итоговое предсказание будет следующим: $\hat{y} = \frac{|L|}{|Q|}\hat{y}_L + \frac{|R|}{|Q|}\hat{y}_R$.

Предсказание решающего дерева можно записать в следующем виде: $\hat{y}(x) = \sum_{i=1}^{N} w_i I[x \in C_i]$ - взвешенная сумма индикаторов принадлежности элемента x к каждому из классов. Этим действи-

ем мы сформировали новое признаковое описание данных, где в качестве признаков выступает принадлежность к конкретному классу. Тогда написанная выше формула всё равно что линейная модель.